DBTaint: Cross-Application Information Flow Tracking via Databases Benjamin Davis, Hao Chen (University of California, Davis)

Motivation

As we become more reliant on Web services, these services become more attractive targets to attackers. Tracking the flow of "tainted" (untrusted) data through a system is a proven method of detecting vulnerabilities and preventing many types of attacks. Unfortunately, most existing information flow tracking systems are a poor fit for the multi-application architecture of modern Web services.

The Problem

Many existing information flow system can only track tainted data through a single application. However, most Web services include at least one Web application and one database application. Single-application systems lose taint information at the application boundaries, leaving users with imperfect options including:

- Consider all data from databases tainted
- Consider all data from databases untainted
- Manual annotation

Application-specific decisions or specialized environments

Many system-wide information flow tracking systems are too coarsely grained (operating at the process or file level), while fine-grained systems that operate at the instruction level lack the ability to make use of high-level database semantics.

DBTaint provides

•End-to-end information flow tracking through Web services, across Web applications and databases

•Mechanisms for leveraging single-application information flow systems in multiapplication Web services

•Information flow tracking in existing Web services, requiring no changes to Web applications

Taint propagation through database functions

Use Cases

- Persistent cross-application information flow tracking
- Regression testing, bug detection
- •Identification of incomplete sanitization policies via column analysis

Implementation

- Perl applications
- Perl DataBase Interface (DBI) API
- Java applications
- Java Database Connectivity (JDBC) API
- PostgreSQL Database

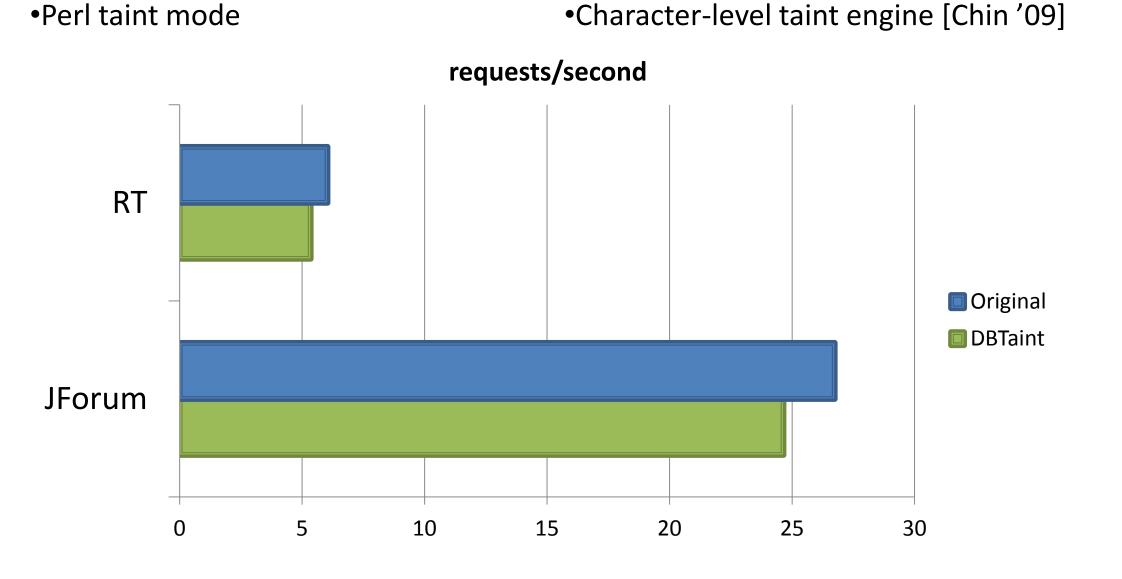
Evaluation

RequestTracker (ticket tracking system) •60,000+ lines of Perl

Perl DBI (DataBase Interface) API

JForum (discussion board system) •30,000+ lines of Java

 Java Database Connectivity (JDBC) API •Character-level taint engine [Chin '09]



Query Rewriting

To propagate taint information from the Web application to the database, DBTaint rewrites SQL queries to include taint values corresponding to the query data values. Rewriting the SQL queries in the database interface means that the augmentations made to database tables to store taint values remain transparent to the Web application.

DBTaint Database Interface

Web App

Processing Query Results The DBTaint modifications in the database interface take the composite results from a database query and collapse them into appropriately tainted values before returning them to the Web application.

Single-Application Information Flow System

DBTaint enables us to leverage existing single-application information flow systems in a multi-application Web service without losing taint data at the application boundaries.

Storing Taint Values

DBTaint stores taint values alongside data values using composite types, which are tuples of the form: (<data value>, <taint value>)

This approach allows DBTaint to use SQL to access and manipulate taint values in the database portably, requiring no changes to the database engine.

Database

Propagating Taint in **Database Functions**

We provide taint-propagating versions of internal database functions, including: Aggregate functions: MAX, MIN •Comparison functions: =, <, !=, ... Helper functions (for composite types): getval(), gettaint()

No Changes to the Web Application

By operating completely in the Database Interface, DBTaint requires no changes to the Web application.